Modeling and Control of Circulating Fluidized Bed using Neural Networks

Electrical Engineering Department
WVU Tech

Advisor:
Dr. Asad Davari

Graduate Students:
Amol Patankar
Praveen Koduru

NETL Mentors:
Dr. Lawrence Shadle
Larry Lawson

Annual Project Report
University/NETL Partnership Program

(Subcontract agreement No. 001060-2
under prime DOE agreement No. DE-FC-26-98FT40143)

November 2001

TABLE OF CONTENTS

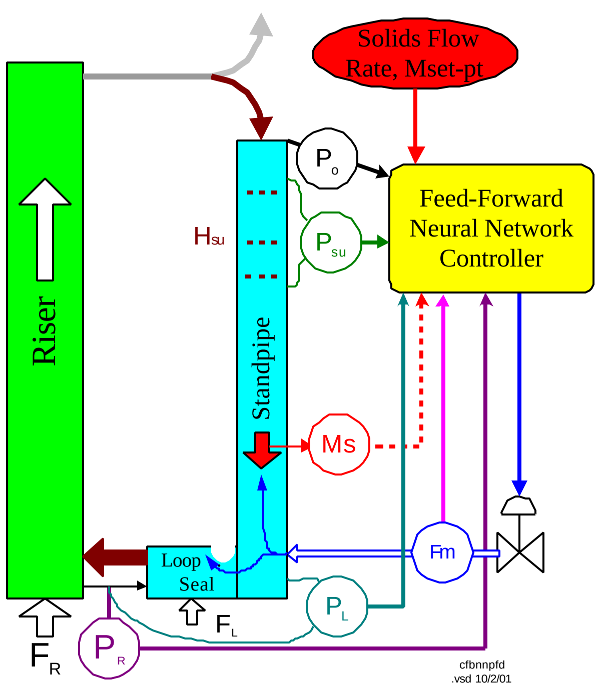
LIST OF GRAPHICAL MATERIAL

1. INTRODUCTION

1.1 CIRCULATING FLUIDIZED BEDS

Circulating Fluidized beds are a relatively new method of forcing chemical reactions to occur in chemical and petroleum industries. A circulating Fluidized bed is a closed system used to mix a solid with a gas in order to force a chemical reaction. Typically in an industrial setting, CFBs implement combustion as part of the process to speed up the rate of reaction. The CFB under consideration is a “cold” circulating fluidized bed, meaning there is no combustion component in its process. The reason for eliminating combustion from this unit is to isolate and study the effects of the internal pressure of the system, independent of temperature effects.

A schematic of the CFB is shown in figure 1. One of the major problems in the study and design of these large complex systems is the modeling and prediction of their characteristic behavior. Currently there is no way to construct a reliable model of such a complex system using traditional methods. Three major obstacles in characterizing the system are:



- Chaotic nature of the system
- Non-linearity of the system

- Number of immeasurable unknowns internal to the system

CFB

Artificial Neural Networks (ANNs) have the ability to characterize the system, and can overcome all three of these obstacles. Neural Networks provide a simple way to construct both of these tools. Neural Networks have the greatest promise in the realm of nonlinear control problems. This stems from their theoretical ability to approximate arbitrary nonlinear mappings.

1.2 NEURAL NETWORKS

Neural networks generally consist of a number of interconnected processing elements or neurons. How the inter-neuron connections are arranged and the nature of the connections determines the structure of the network. Its learning algorithm governs how the strength of the connections are adjusted or trained to achieve a desired overall behavior of the network. A simple feed forward neural network is shown in Figure 2.

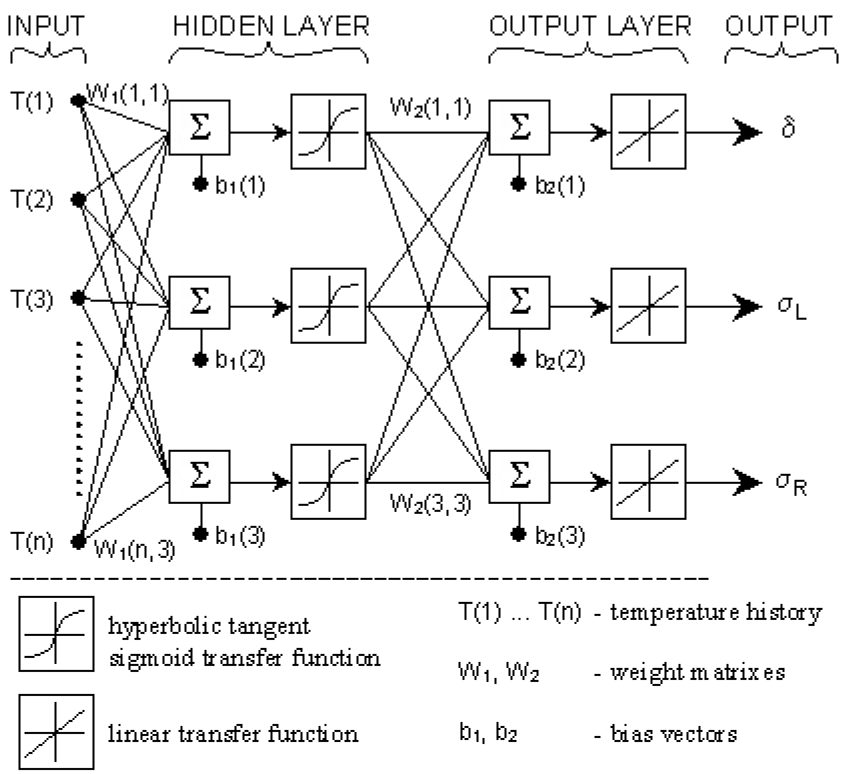


Figure 2. Representation of a Feed forward Neural Network

In feed forward network, the neurons are generally grouped into layers. Signals flow from the input layer through to the output layer via unidirectional connections, the neurons are being connected from one layer to the next, but not within the same layer. Between the input layer and the output layer, we have the hidden layers. As the complexity of the problem being solved increases, the size and number of the hidden layer increases. A supervised learning algorithm adjusts the strengths or weights of the inter-neuron connections according to the difference between the desired and actual network outputs corresponding to a given input. Thus, supervised learning requires a "teacher" or "supervisor" to provide desired or target output signals.

1.2.1. Backpropagation Algorithm

An example of such supervised learning algorithm is Backpropagation (BP) algorithm.

Multi-Layer Perceptrons (MLP) are perhaps the best-known type of feed forward networks. MLP has generally three layers: an input layer, an output layer and an intermediate or hidden layer. Neurons in the input layer only act as buffers for distributing the input signal x_i to neurons in the hidden layer. Each neuron j in the hidden layer sums up its input signals x_i after weighting them with the strengths of the respective connections w_{ji} from the input layer and computes its outputs y_j as a function f of the sum, viz.

$$y_j=f(\Sigma w_{ji} x_i) \qquad (1)$$

f can be a simple threshold function or a sigmoid, hyperbolic tangent or radial basis function.

The output of neurons in the output layer is computed similarly. The BP algorithm, a gradient descent algorithm, is the most commonly adopted MLP training algorithm. It gives the change Δw_{ji} in the weight of a connection between neurons j and i as follows:

$$\Delta w_{ji} = \eta \delta_j x_i \qquad (2)$$

Where η is a parameter called the learning rate and δ_j is a factor depending on whether neuron j is an output neuron or a hidden neuron. For output neurons,

$$\delta_j = \left(\frac{\partial f}{\partial net_j} \right) \left(y_j^{(t)} - y_j \right) \qquad (3)$$

and for hidden neurons,

$$\delta_j = \left(\frac{\partial f}{\partial net_j} \right) \sum_q w_{qj} d_q \qquad (4)$$

In equation (3), net_j is the total weighted sum of input signals to neuron j and $y_j^{(t)}$ is the target output of neuron j.

As there are no target outputs for hidden neurons, in equation (4), the difference between the target and actual output of a hidden neuron j is replaced by the weighted sum of the δ_q terms already obtained for neurons q connected to the output of j. Thus, iteratively, beginning with the output layer, the δ term is computed for neurons in all layers and weight updates determined for all connections. The steps have been clearly illustrated in Figure 3.

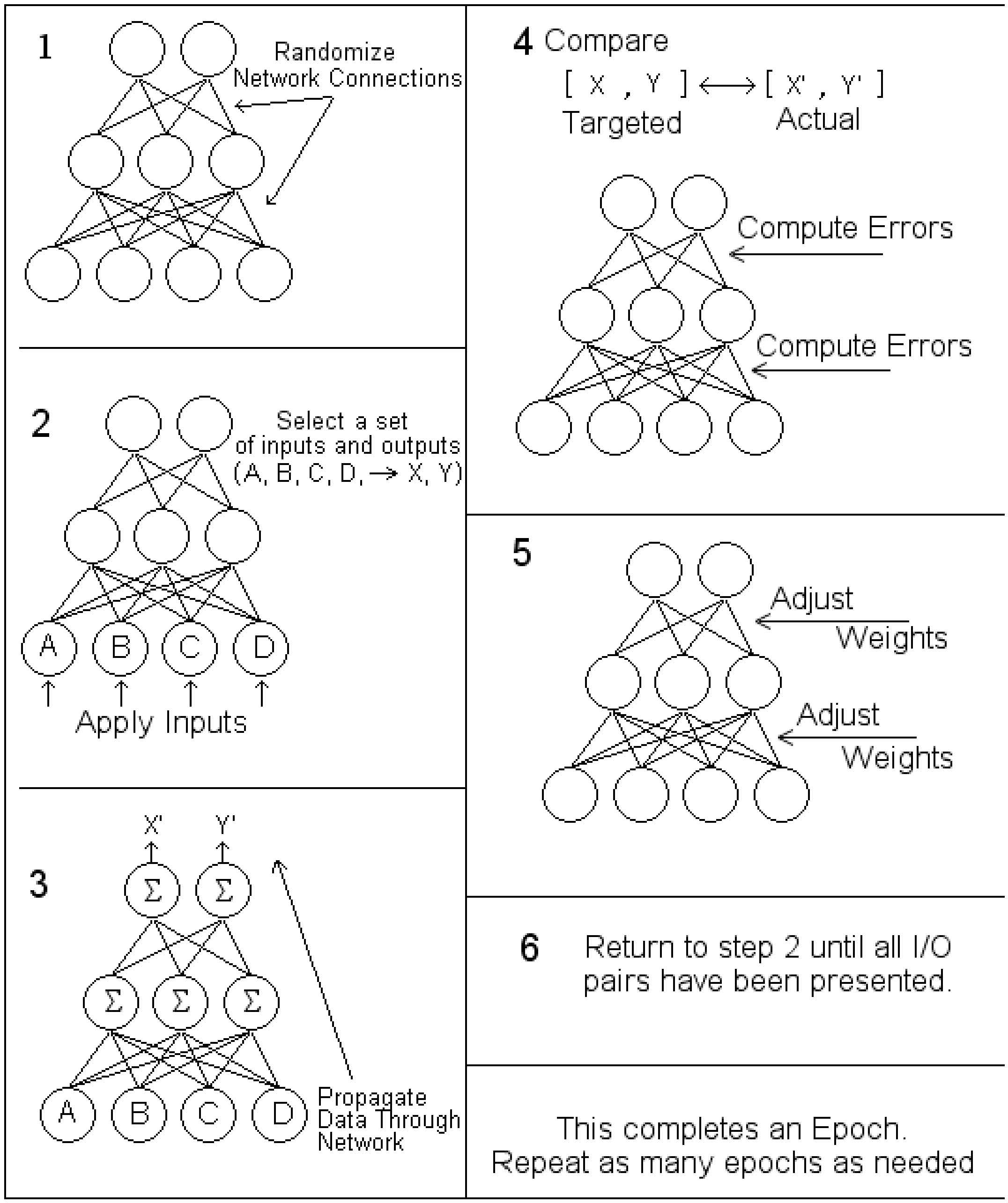


Figure 3. Backpropagation Algorithm

1.2.2. Backpropagation Algorithm: The Levenberg-Marquardt (LM) Method

The LM method, a second order optimization method, was used to determine the weights in the NN. Network training aims at minimizing the sum of squares of errors, the errors measured as the difference between the calculated output and the desired output. Minimizing the quadratic error is not always the best way of training a NN, but for this application, it suffices. Most algorithms for least-square optimization problems use either steepest descent or Taylor-series models. The Levenberg-Marquardt method uses an interpolation between the approaches based on the maximum neighborhood (a "trust region") in which the truncated Taylor series gives an adequate representation of the nonlinear model. The method has been found to be advantageous compared to other methods, which use only one of the two approaches. This makes LM training much faster than gradient descent. For networks that are not explicitly recurrent LM training is very good method.

The equations for changing the weights during training are given as follows:

$e_k = O(t_k) - \tilde{O}(t_k)$ where $\tilde{O}(t_k)$ is the neural network output.

$$J_{kl} = \left.\frac{\partial O}{\partial W_l}\right|_{tk}$$

$$O \cong \vec{e} + \frac{\partial \vec{e}}{\partial \vec{W}} \Delta \vec{W} = \vec{e} - J \Delta \vec{W}$$

$$\Delta \vec{W} = (J^T J)^{-1} J^T \vec{e}$$

$$Modifying \Rightarrow \Delta \vec{W} = (J^T J + mI)^{-1} J^T \vec{e} \qquad (5)$$

1.3. BLACK-BOX MODELING APPROACH

These is the most common form of supervised learning, giving the NN, off-line data of the plant, and mapping the correlation between the inputs and outputs, forming a black-box model of the plant as shown in Fig 4. In the system under consideration, we have tried to model the mass flow rate as a function of the differential pressures in the standpipe section and the flow rate of the aeration. The ultimate aim of the work is to develop a neural network controller to manipulate the mass flow rate by varying the aeration.

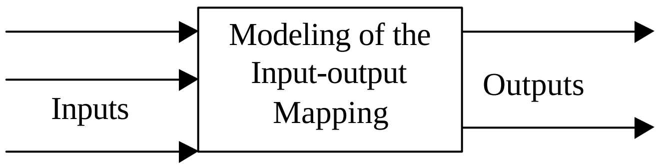


Figure 4. Schematic of Black Box modeling

Figure 5 shows the schematic of Black Box modeling approach as implemented on the present CFB.

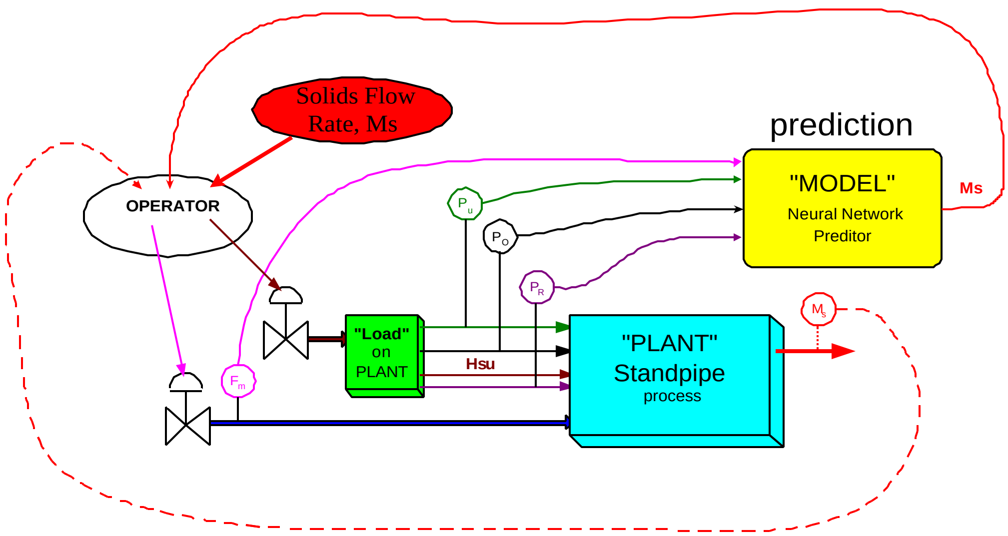


Figure 5. Black Box Modeling of CFB

The tricky part in modeling the system was the collection of data to span over the entire operating range. The data points had to be chosen in such a way that all the system dynamics were captured. Figure 6 shows a sample set of data variables collected, showing the variation of each variable over that period. Note that all variables have been scaled between 0 and 1 to fit on same graph.

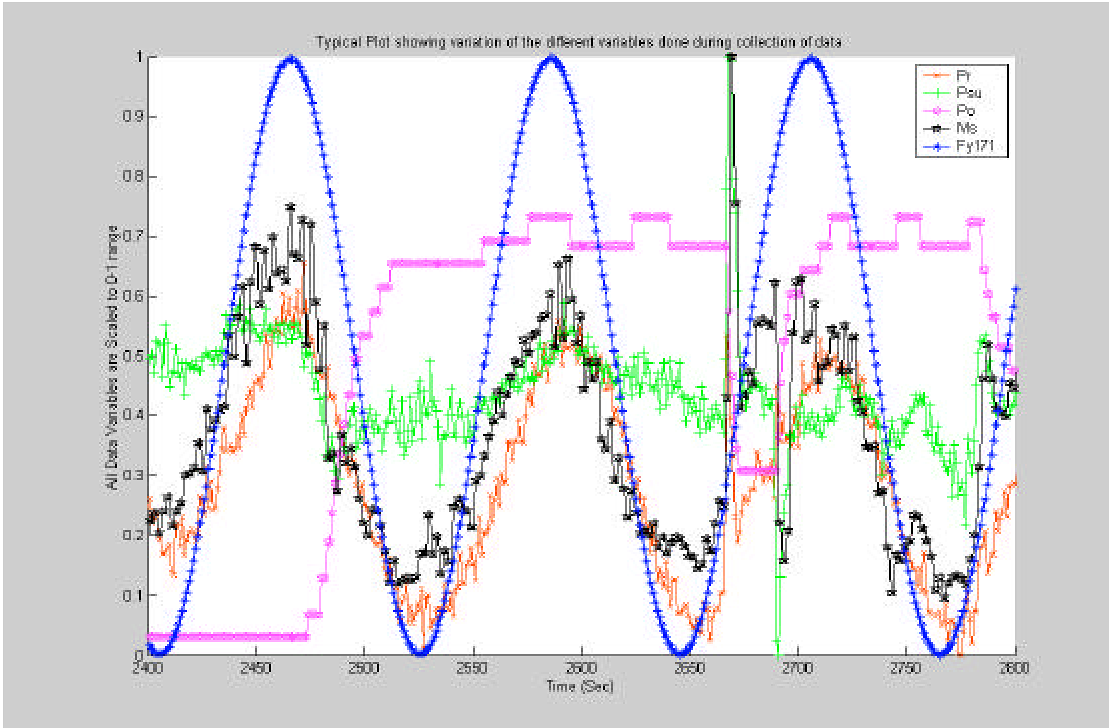


Figure 6. Sample set of scaled data points

Out of the 3600 data points collected, 1800 points were used for modeling and the remaining points were used for extrapolation of the results over unseen data points for validation of the obtained model.

Determining the size of the layers is an important issue. One of the most used approaches is the constructive method. Constructive methods determine the topology of the network during the training phase as an integral part of the learning algorithm. The common strategy of the constructive methods is to start with a small network, train the network until the performance criterion has been reached, add a new node and continue until a 'global' performance in terms of error criterion has reached an acceptable level.

The expected benefits using such algorithms are:

- Trial-and-Error: The usual trial and error process for finding the 'best' topology is avoided.

- Speed: Because of fewer free parameters the network is expected to converge faster.

The main objective of this work is to train the neural network model to provide a suitable off-line model that simulates CFB operation. If the neural network plant model is capable of approximating with sufficient accuracy the highly complex process in the CFB, it may be used within a model based control strategy.

2. MODELING THE CFB

In the system under consideration, we have modeled the mass flow rate as a function of the differential pressures in the standpipe section and the flow rate of the aeration. The ultimate aim is to develop a neural network controller to manipulate the mass flow rate by varying the aeration. A P & I diagram of the CFB is shown in Figure 7.

The major variables used for modeling and their range are as follows:

PDT 841 PDT 842 PDT 811A PDT 801	Riser Pressure (Pr)	Range 0.3 – 0.7 psid
PDT 853 PDT 864	Upper Stand pipe pressure (Psu)	Range 0 – 0.4 psid
PIC 941	Back pressure (Po)	Range 0 – 2.6 psig
F874	Mass flow rate (Ms)	Range 600 – 6000 lb/hr
FY 171	Aeration rate (Fm)	Range 300 – 700 scfh

We have also incorporated the time-delayed signals as FY171 (t-1) and FY 171 (t-2) to enable capturing the delayed effects of respective variables as FY 171 on the system.

Here all the PDTs and PIC represent the differential pressures in the standpipe, F874 represents the mass flow rate and FY 171 represents the aeration.

To model the system we have used multi layered neural network with neurons in the input layer, hidden layer and the output layer. We have used the Levenberg Marquardt (LM) algorithm for training the neural network. The training was done using the Neural Network toolbox in Matlab. The tricky part in modeling the system was the collection of data to span over the entire operating range. Data was collected by varying move air (aeration) in a sinusoidal manner over its whole range. All other variables were also being manipulated manually over their respective range. The data points had to be chosen in such a way that all the system dynamics were captured. Out of the 3600 data points collected, 1800 points were used for modeling and the remaining points were used for extrapolation of the results over unseen data points for validation of the obtained model. The blue signal in the plots is the plant response and the red signal is the neural network output. The first 1800 points represent the data points used for training while the remaining points were used for validation of the model.

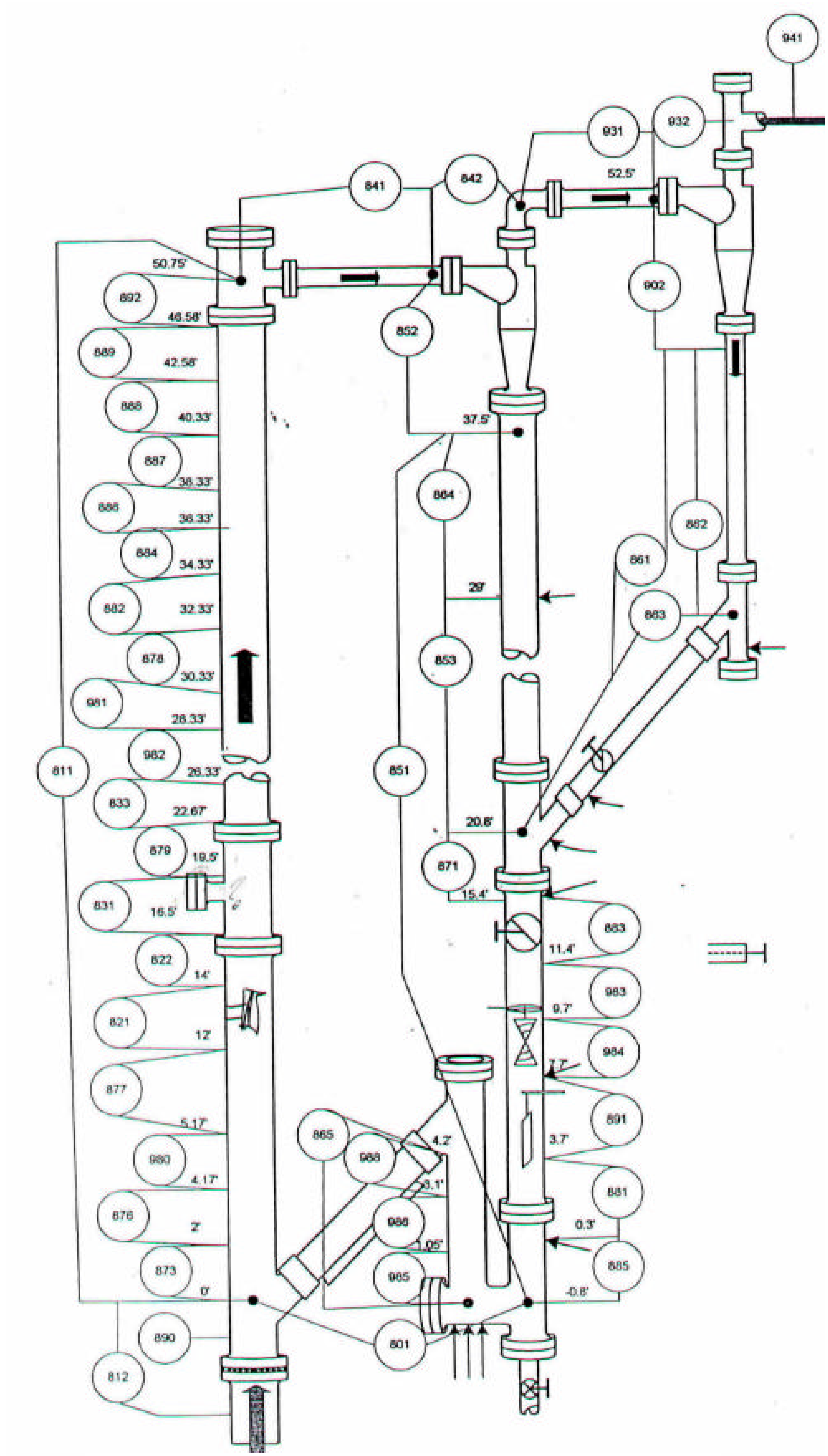


Figure 7. P & I diagram of the CFB.

3. RESULTS AND DISCUSSION

3.1. MODELING

Two related problems that must be addressed in training a NN are: how to determine the architecture (how many neurons and layers) and how to avoid over fitting. Since NN is highly redundant and over-parameterized, it is easy to fit the noise in the data as well as signal. To avoid this problem, the standard NN modeling practice is to separate the data into a "training set", which is used to train the model, and a "testing set", which is used to determine when to stop training. Only the training set is used to adjust the network weights. During training, the model is evaluated periodically using the training set. A decrease in testing performance signals that over fitting is setting in. In addition, a "validation set" may be set aside for evaluating the model on data that was not involved in the training process. We have also incorporated the time-delayed signals of the

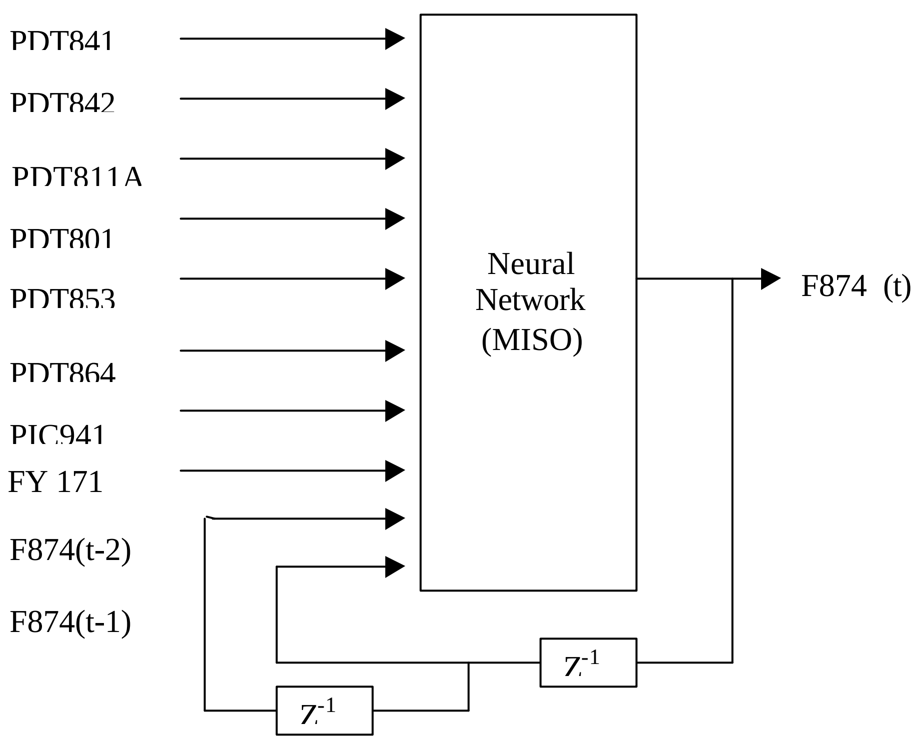


Figure 8. MISO Neural Network

predicted variable to enable capturing the delayed effects of that variable on the system. Also normalizing the variable within the range of [0,1], has given better results with a smaller network and better approximation.

3.1.1 Multi Input Single Output (MISO) Neural Network Model

A schematic of the MISO neural network model for the prediction of the Mass flow rate (Ms) is shown in Figure 12.

The training and the extrapolation data for the two most dominating variables of the CFB, the Mass flow rate and the Aeration are being presented in Tables 1 and 2. The value of Mean Squared Error (MSE) for both the training set and the Test set indicate the optimal network for the process variable and also help in reducing the chances of over-fitting the data. The Network Size represents the number of hidden layers and the number of neurons in the hidden layer.

Table 1. Training and Testing data of Mass Flow Rate (M_s).

Network Size	Training Algorithm	Activation Function	Training Time (min)	No. of Iterations	MSE Error	MSE Error
					Training	*Test*
5	LM	Tansig	0.3160	25	0.0021	0.0034
10	“	Tansig	0.3538	“	0.0019	0.0034
“	“	“	0.4271	50	0.0018	0.0033
“	“	Logsig	0.3544	25	0.0021	0.0036
15	**“**	***Tansig***	***0.3710***	**“**	***0.0021***	***0.0032***
”	“	“	0.4822	50	0.0019	0.0034
25	“	“	0.4428	25	0.0020	0.0034
25	GD	Tansig	0.3266	“	0.0291	0.0321

Table 2. Training and Testing data of Aeration (Fy).

Network Size	Training Algorithm	Activation Function	Training Time (min)	No. of Iterations	MSE Error	MSE Error
					Training	*Test*
5	LM	Tansig	0.3392	25	9.86e-6	2.55e-5
10	“	Tansig	0.3744	“	2.12e-5	6.50e-5
“	“	“	0.5403	50	8.94e-6	1.63e-5
“	“	Logsig	0.5011	25	0.0267	0.0275
15	“	Tansig	0.4698	“	9.39e-6	1.49e-5
“	**“**	**“**	***0.6903***	***50***	***9.19e-6***	***1.33e-5***
25	”	“	0.6913	25	3.46e-5	1.12e-4
25	GD	“	0.3435	“	0.1224	0.1226

It is clearly evident from Table 1 and Table 2, the optimal NN for the modeling of the Mass flow rate and Aeration Rate is a single hidden layer of 15 neurons and “Tansig” as the activation function which is represented in bold. Aeration rate needed more number of iterations for better prediction, so as to minimize the error in the Test data. Note from the Tables, the “logsig” activation function was not able to do a good job here. And also we see that the LM training algorithm is superior to the Gradient Descent (GD) training algorithm previously used in modeling. In Figure 11, a magnified view of the extrapolation is shown wherein the variables have been varied over the short range by manipulating the back pressure Po thereby affecting riser pressure, which in turn affects the mass flow rate. The network extrapolates the mass flow rate correctly even in this interacting condition proving that it has correctly learnt this intricate relationship between variables. The plots of the training and the extrapolation of Mass Flow Rate, Ms (F874) are presented in the Figures 9 & 10. The plots clearly show that the Neural Network model has not only learnt the system dynamics but was also able to extrapolate over unseen data, which was an acid test for the validity of the model. Only the data from 1-1800 sec was given for the training, and the rest of the data was unseen by the Neural Network during training. It was only in the testing phase that data was being used to test the Trained Network.

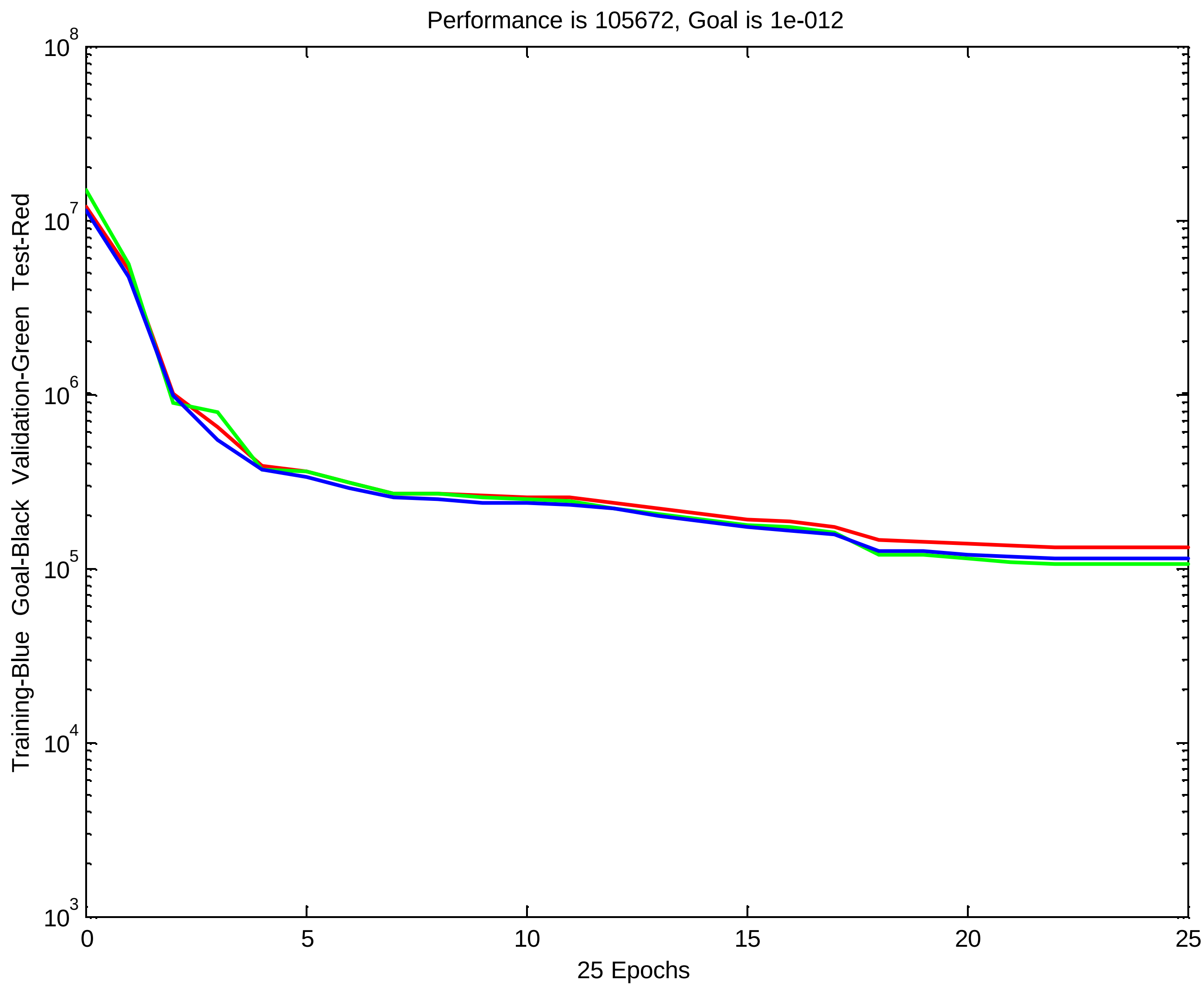


Figure 9. Mean Squared Error (MSE) plot during training of Mass Flow rate (Ms) for MISO model

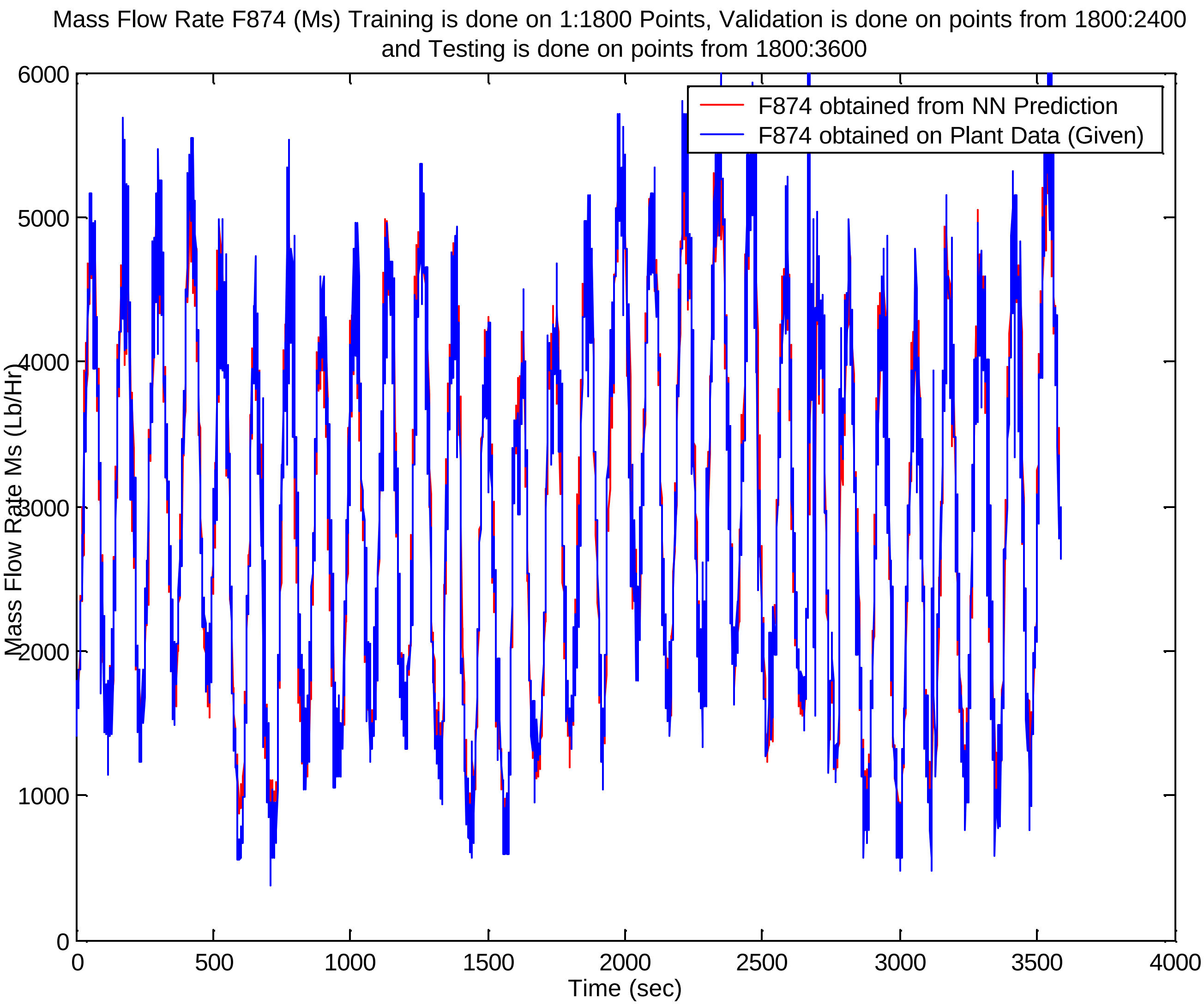


Figure 10. Plot showing prediction of Ms by the trained Neural Network MISO Model

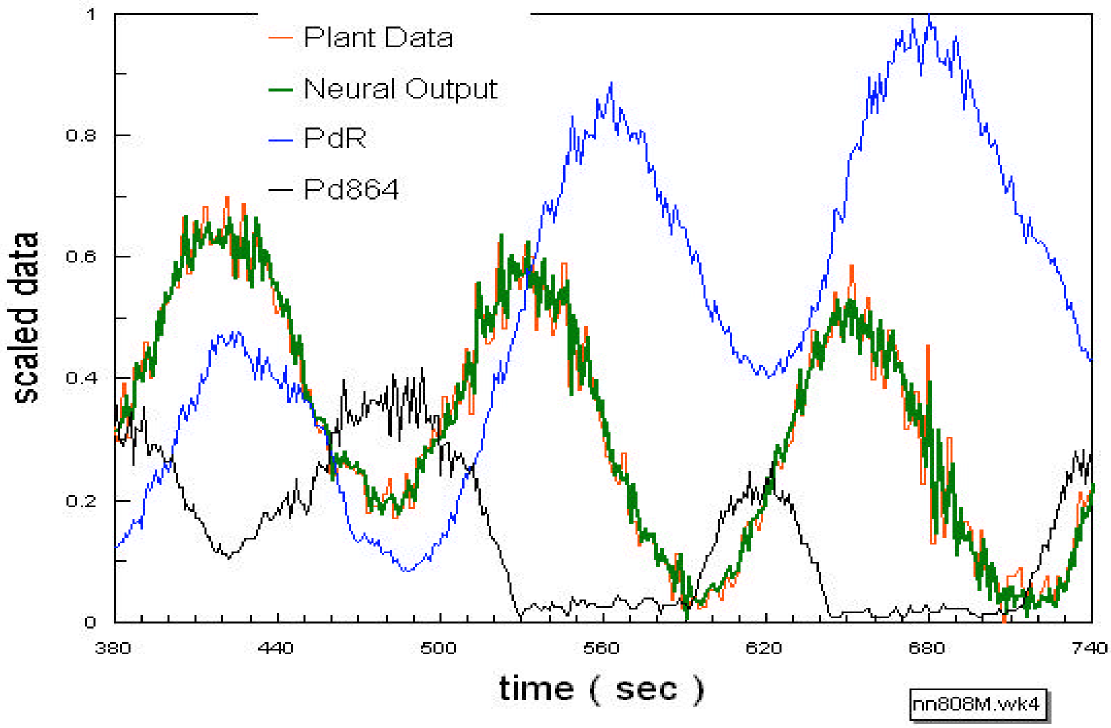


Figure 11. Neural network Extrapolation of Mass Flow rate (Ms)

3.1.2. Multi Input Multi Output (MIMO) Neural Network model

A schematic of the Multi Input Multi Output Neural Network model is shown in Figure

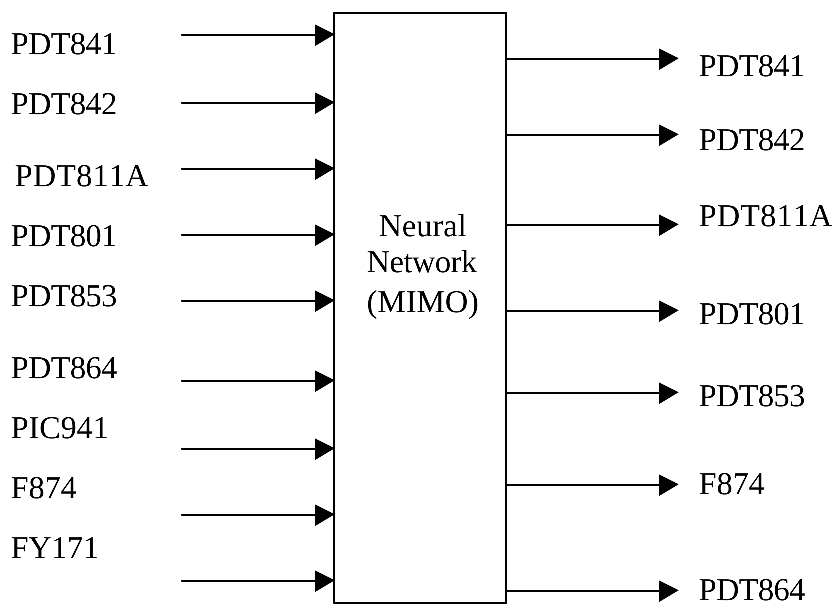


Figure 12. MIMO Neural Network

12. The inputs are at time 't' and the NN predicts outputs for time 't+1'. The plots showing the extrapolation by the MIMO Neural Network model are shown in the Figures 13 through 19.

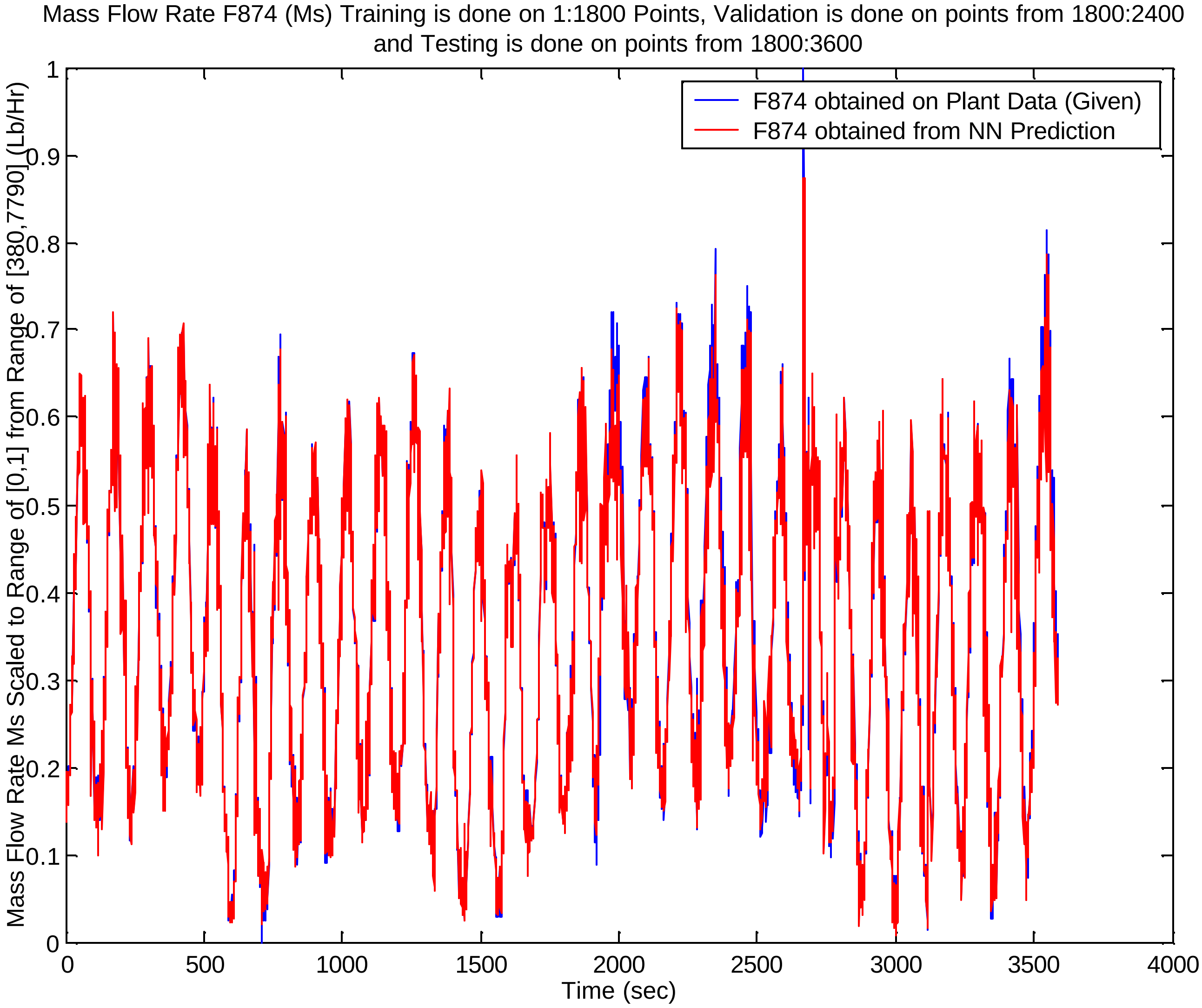


Figure 13. Plot showing prediction of Ms by the trained Neural Network MIMO Model

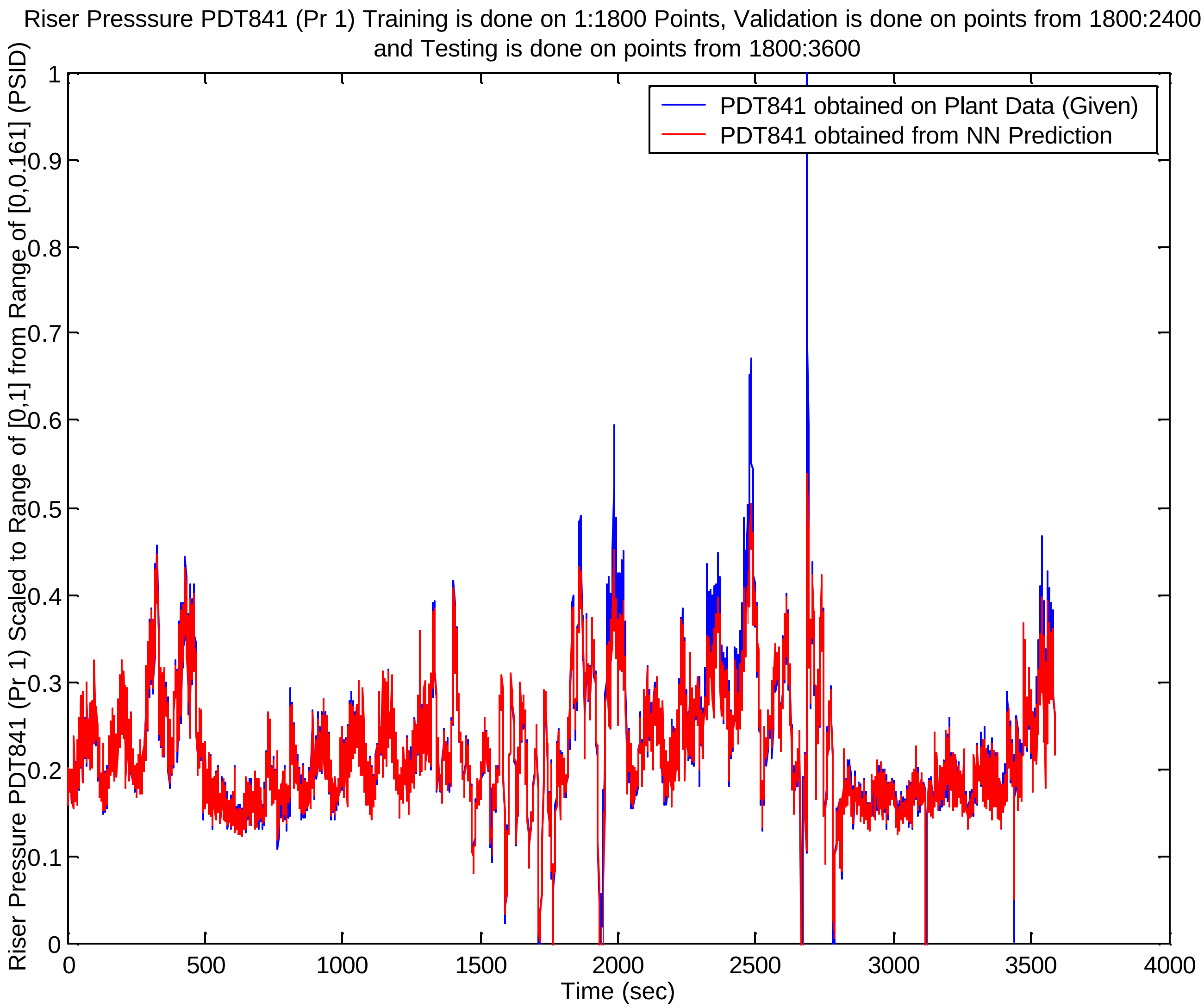


Figure 14. Plot showing prediction of PDT 841 by the trained Neural Network MIMO Model

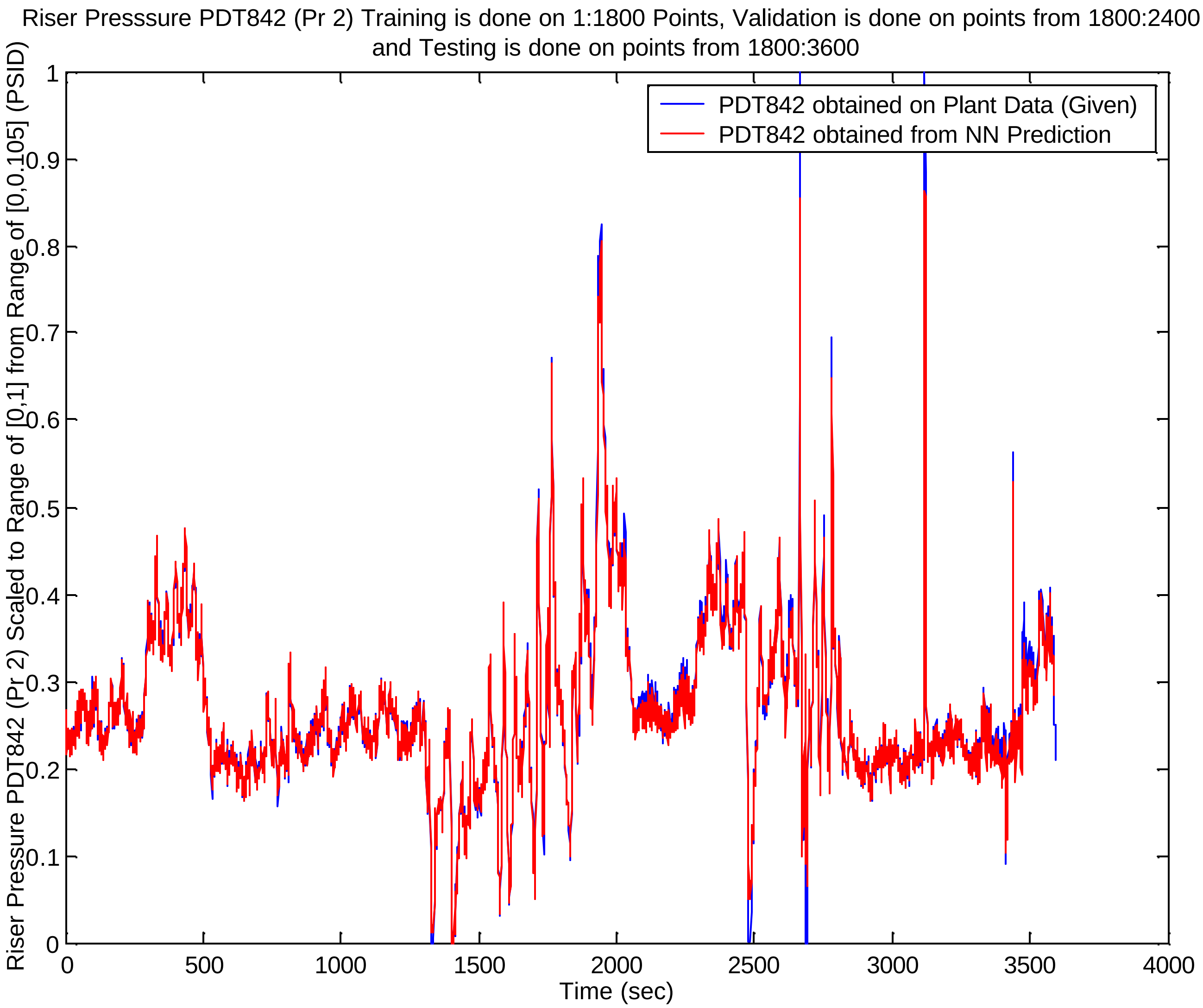


Figure 15. Plot showing prediction of PDT 842 by the trained Neural Network MIMO Model

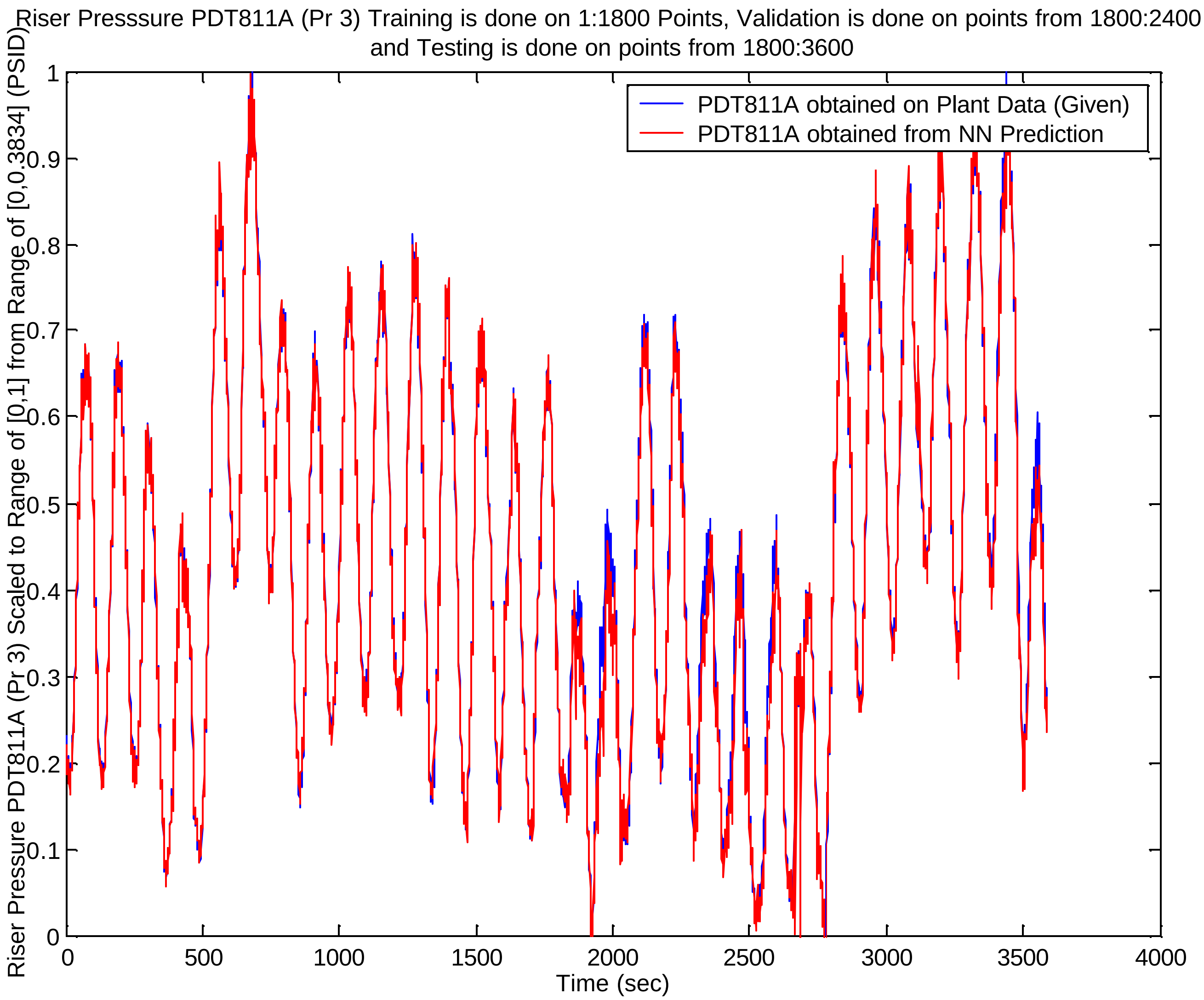


Figure 16. Plot showing prediction of PDT 811A by the trained Neural Network MIMO Model

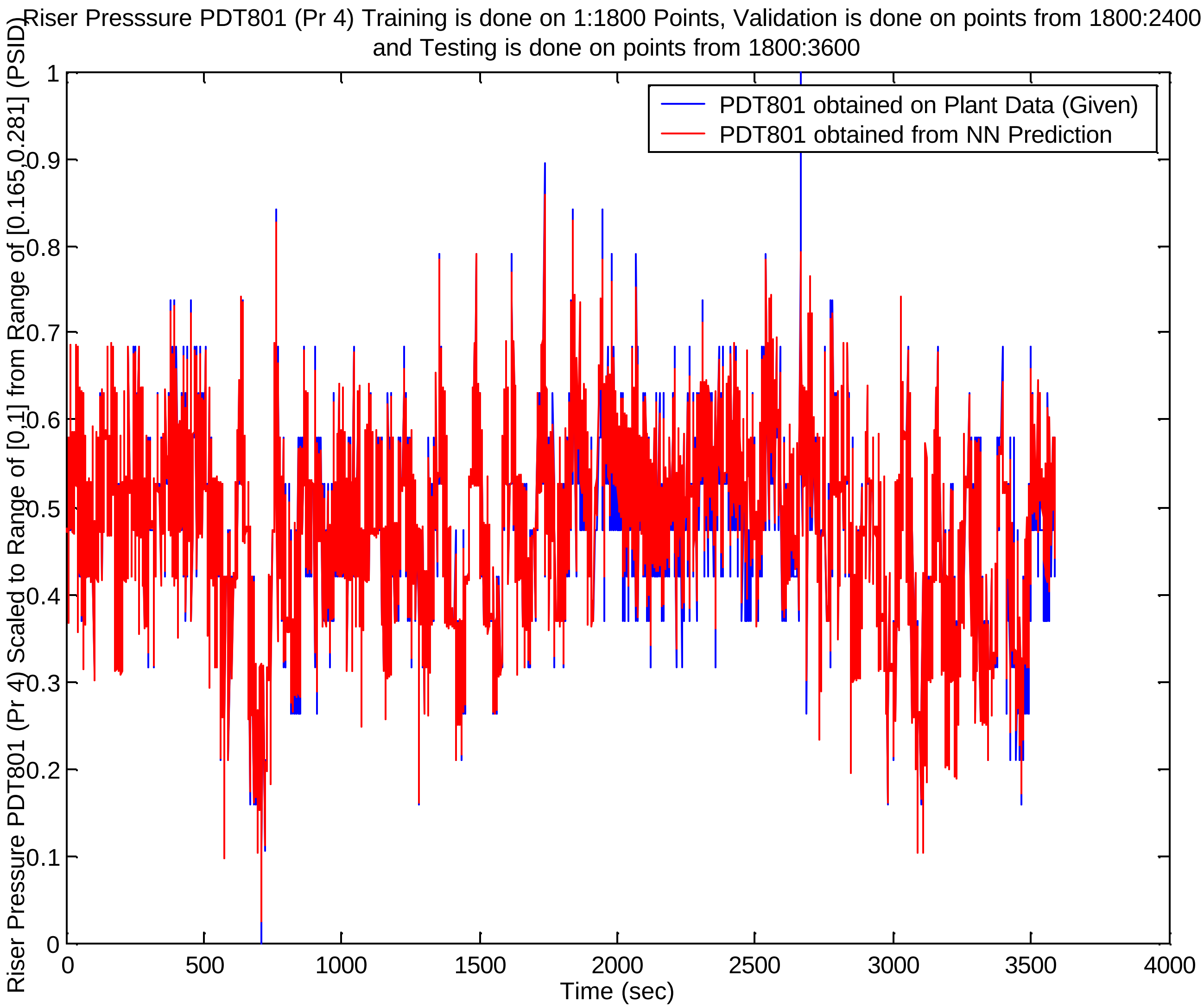


Figure 17. Plot showing prediction of PDT 801 by the trained Neural Network MIMO Model

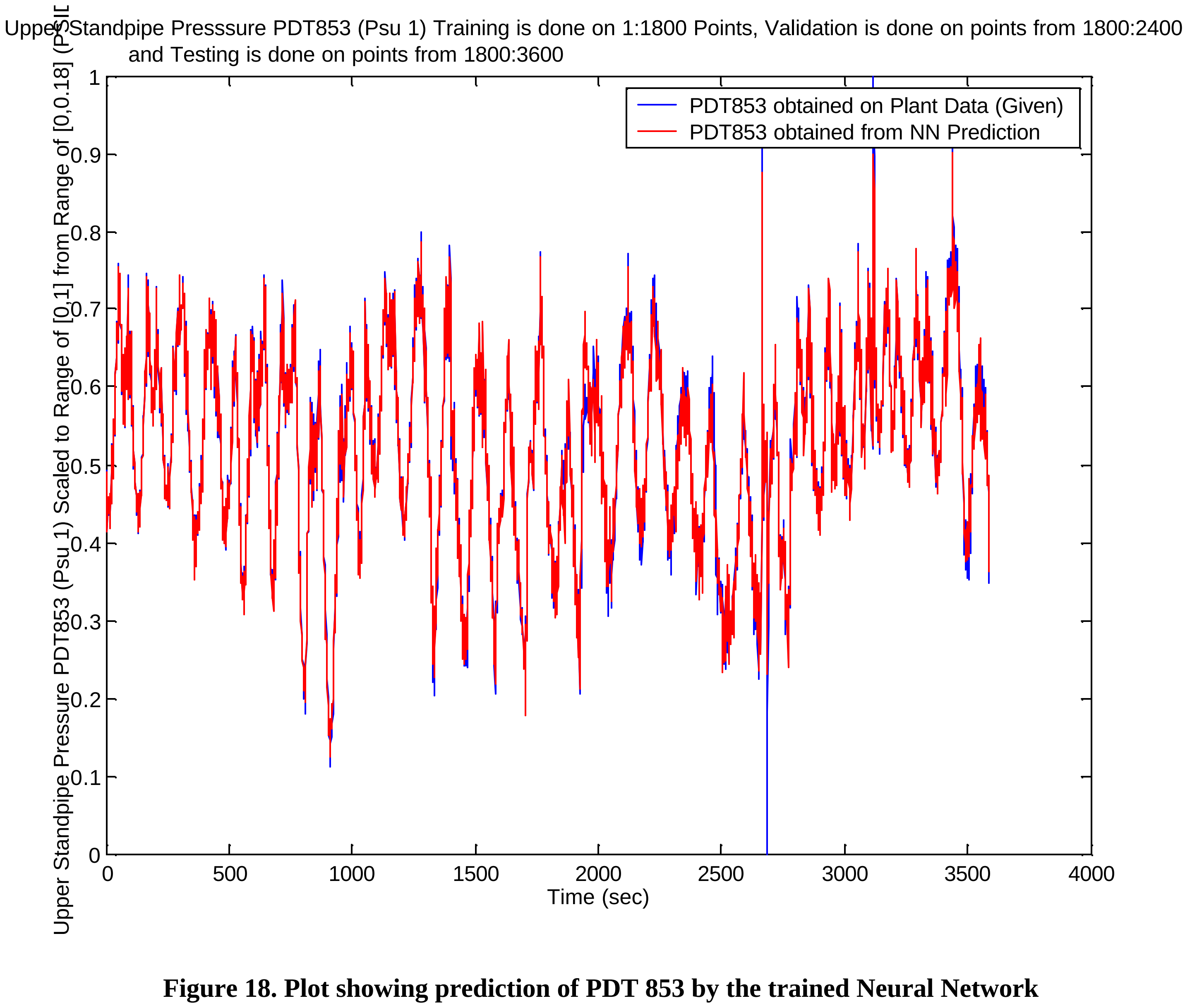


Figure 18. Plot showing prediction of PDT 853 by the trained Neural Network MIMO Model

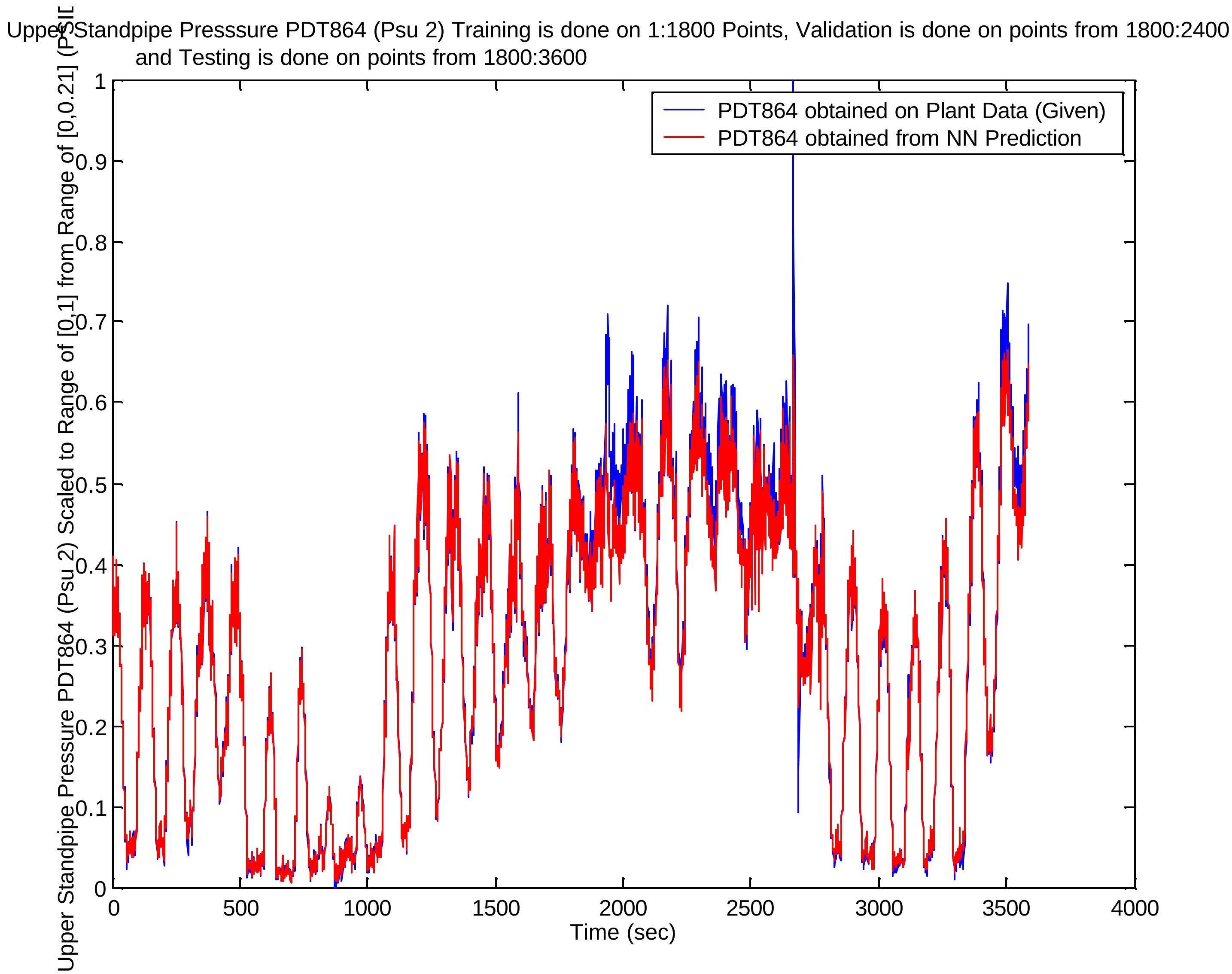


Figure 19. Plot showing prediction of PDT 864 by the trained Neural Network MIMO Model

3.2. DEVELOPMENT OF CONTROLLER

3.2.1. Inverse Model Control

Consider a system transfer function $\frac{Y(s)}{X(s)}$ where Y(s) is the system output and X(s) is the input to the system. The significance of the transfer characteristics in system and control theory is based on the fact that response of a static system to an arbitrary excitation can be computed as a product of its transfer characteristics and of the excitation. Similarly, the transform of the response of a dynamical system can be expressed as a product of the transfer function and transform of the excitation.

For a plant that is completely known, any desired relationship between input and

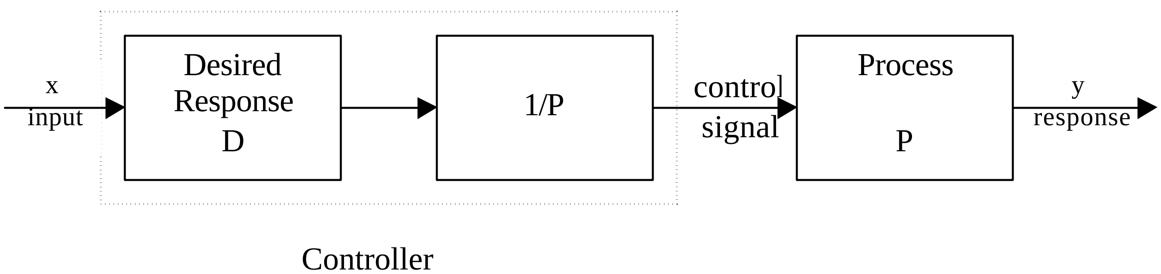


Figure 20. Control of Process P to achieve the desired response.

response can be realized by a simple open loop control configuration as shown in Figure 20. The feed forward controller consists of two parts connected in cascade. The second one cancels the process transfer function because its own transfer function has been chosen as the inverse of the known process transfer function. In addition, the front part of the controller introduces the desired overall transfer function of the controller and the plant equal to D. Thus the controller's transfer function should be equal to D/P . This approach is known as inverse control since the most essential part of the controller should provide the inverse of the plants transfer function. The transfer function of the inverse feed forward control system from the figure computed between its input and output becomes $D(s) = \frac{Y(s)}{X(s)}$. In this approach involving the cascade connection of the controller driving the plant and involving no feedback, the neural controller that acts as the plant inverse needs to be designed.

For the system under consideration, the inverse model was derived from a neural network that predicted the value of the aeration required to produce a desired mass flow rate. This predicted value of aeration was used to control the mass flow rate to a desired set-point value.

To model the system we have used a 3-layered neural network with 10 neurons in the input layer, 15 neurons in the hidden layer and 1 neuron in the output layer. We have used the Levenberg Marquardt (LM) algorithm for training purposes. The data points had

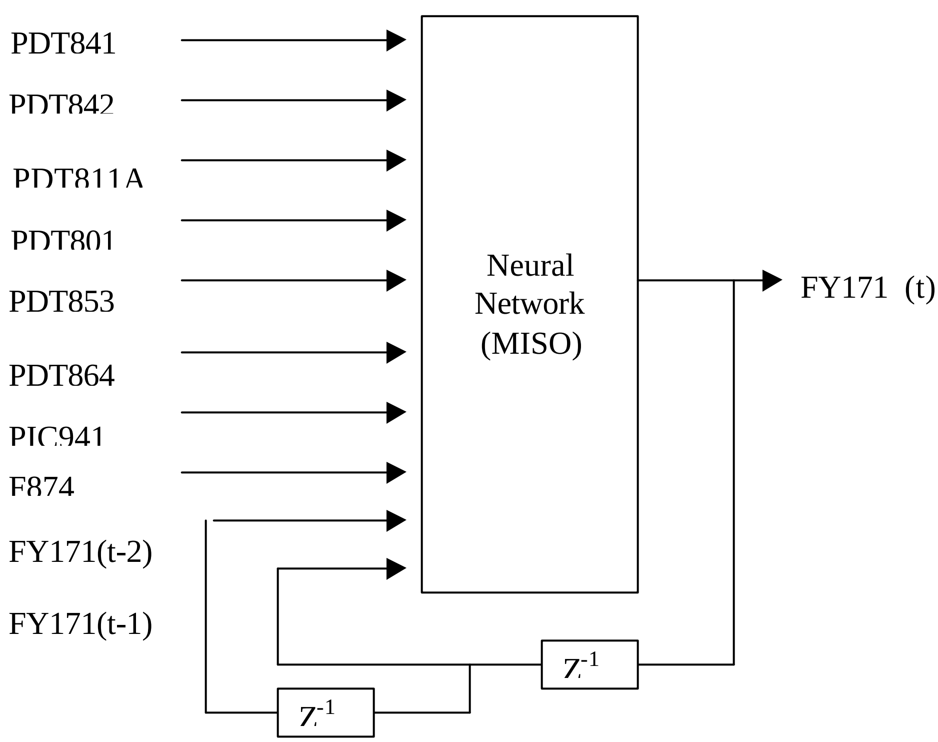


Figure 21. MISO Neural Network for Controller

to be chosen in such a way that all the system dynamics were captured. Out of the 3600 data points collected, 1800 points were used for modeling and the remaining points were used for extrapolation of the results over unseen data points for validation of the obtained model. The trained controller is implemented as an Inverse model of the plant which is explained later. Detailed plots showing the training of the Neural Network controller over the data set being provided and its extrapolation are being shown in Figures 22 & 23.

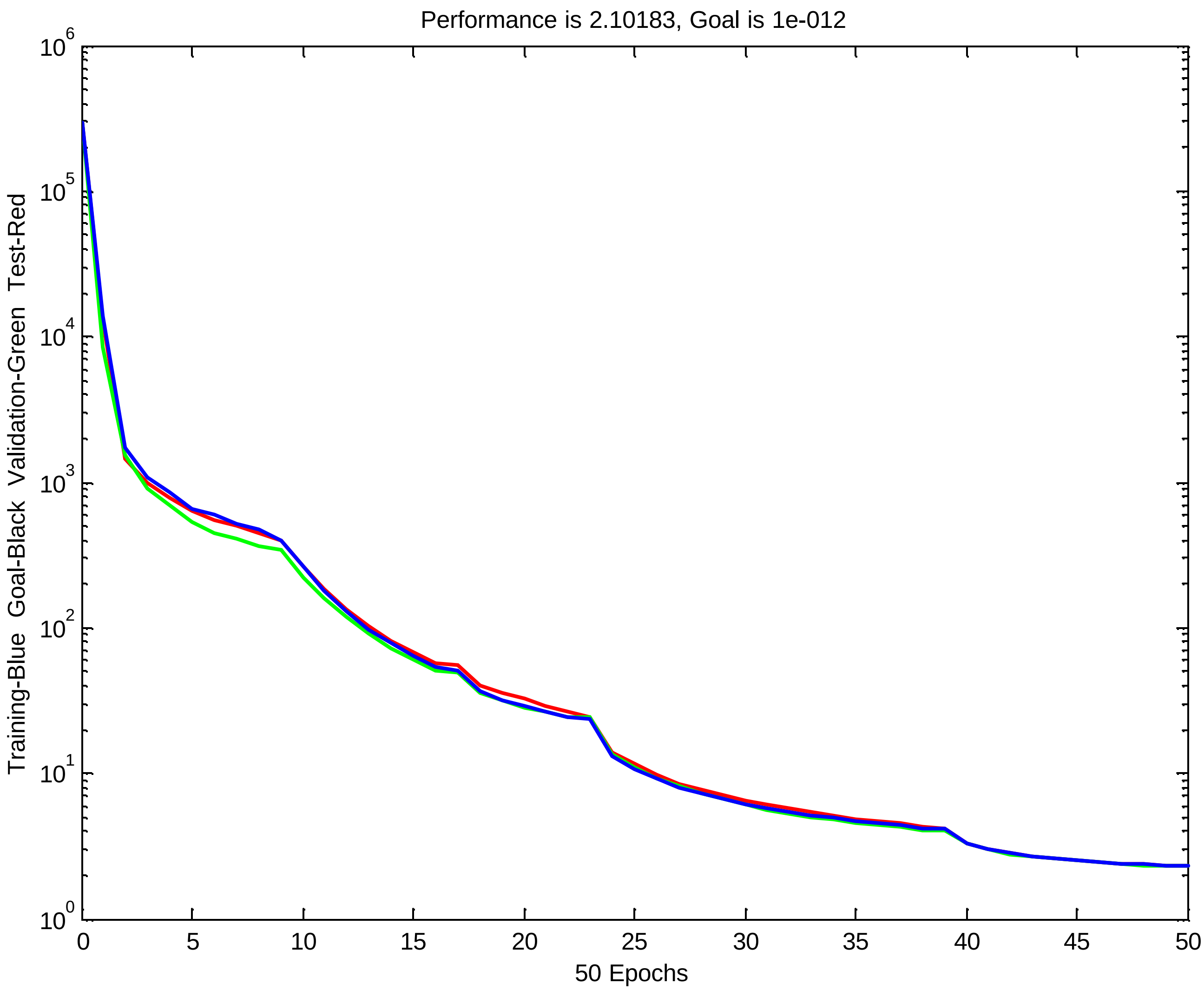


Figure 22. Mean Squared Error (MSE) plot during training of Controller

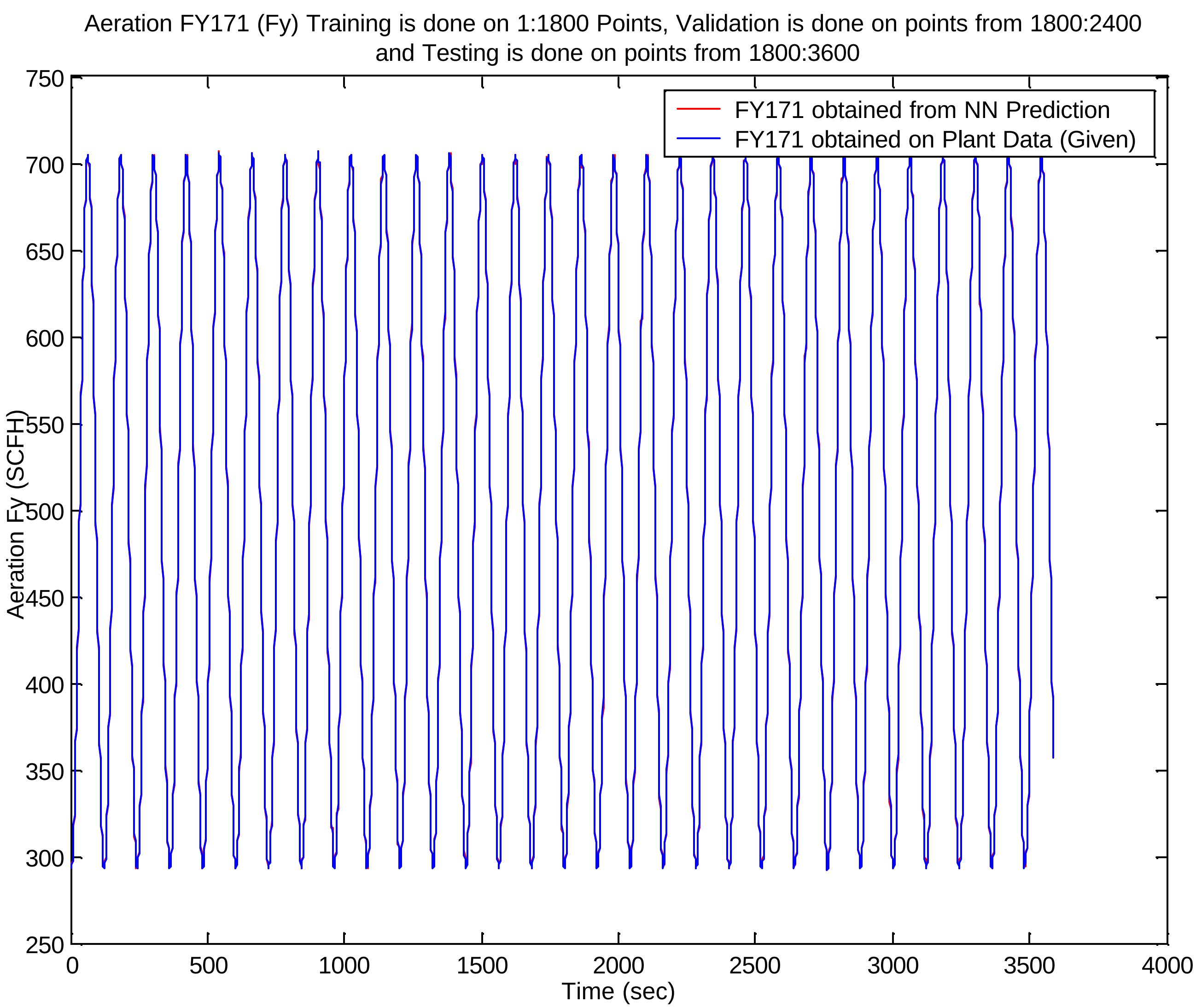


Figure 23. Plot showing prediction of FY 171 by the trained Neural Network Controller

3.2.2. Controller implementation

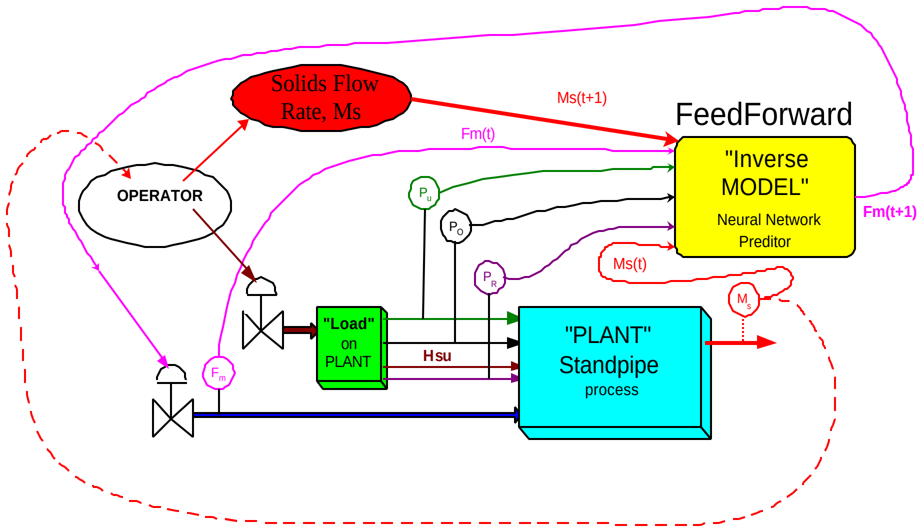


Figure 24. Schematic of Implementation of NN controller on the CFB

We have implemented a feed forward inverse controller where the neural network predicts the values of the aeration rate (FY 171) to achieve a desired value of the mass flow rate as shown in Figure 24. We tested the controller and found it to have a stabilizing effect on the plant. A simulation of this controller is shown in Figure 25.

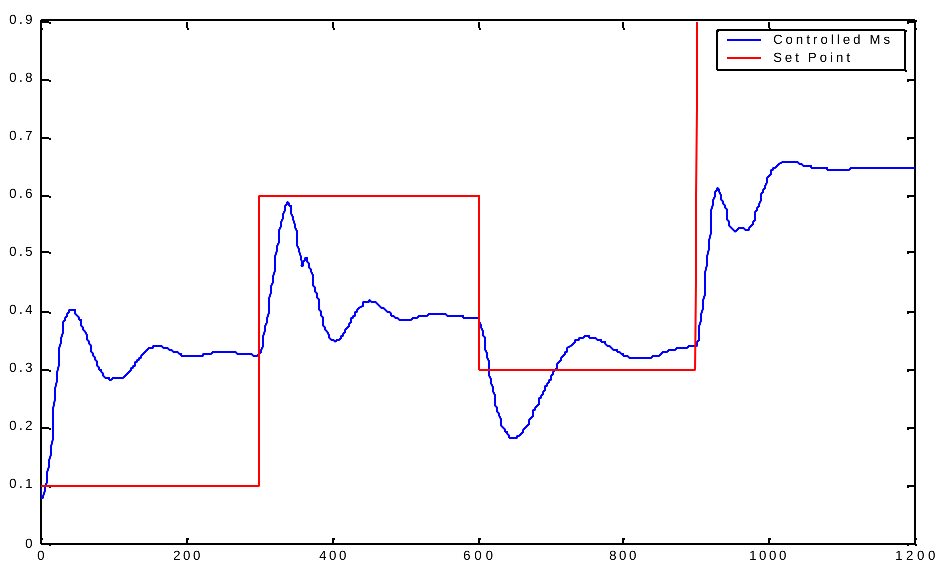


Figure 25. Controller Simulation

4. CONCLUSIONS

With the use of efficient back propagation training algorithm, we were able to make an efficient and accurate model of the CFB. Previous attempts were restricted to the data set and failed on extrapolating the results over unseen data, where we have succeeded. Equipped with an accurate model, the next step is to design and implement an appropriate controller for the CFB.

We are now working on various techniques of implementing the controller, like Internal model control that incorporates feedback signal and relays information about the amount of correction necessary for the control signal to achieve more accurate control. Also, efforts are underway to make the model adaptive to any changes in the plant dynamics.

* * *

4. REFERENCES

1. Daniel Vandel, Asad Davari and Parviz Famouri, " Modeling of Fluidized bed with Neural Networks", Proceedings of 32^{nd} IEEE SSST, March 2000, FAMU-FSU Tallahassee, Florida.

2. Asad Davari, Sridhar Macha and Rammohan Sankar, "Neural Network predictor of a circulating fluidized bed", Proceedings of 33^{rd} IEEE SSST, March 2000, Ohio Univ, Athens, Ohio.

3. Asad Davari, Sridhar Macha and Rammohan Sankar, "Improved Neural Networks modeling and predicting of circulating Fluidized Beds", Proceedings of the IASTED Intl. Conf., May 2001, Pittsburgh, Pennsylvania.

4. A. Davari, and D. Vandel, " Modeling and Predicting the Output of a Circulating Fluidized Bed with Neural Network," Technical Report, DOE/NETL October 1999.

5. A. Davari, and R. Sankar, "Neural Network Predictor of the Output of a Circulating Fluidized Bed," Technical Report, DOE/NETL November 2000.

6. A. Davari, and S. Macha, " Improved Model of Circulating Fluidizes Bed with Neural Networks," Technical Report, DOE/NETL November 2000

7. Levenberg, K., " A method for the solution of certain nonlinear problems in least squares", Quart. Appl. Math., n 2, pp164-168, 1944.

8. Marquardt, D.W., " An algorithm for least-squares estimation of nonlinear parameters", J. Soc. Indust. Appl. Math., n 11, pp 431-441, 1963.

9. Narendra, K.S. and Parthasarathy, K., “Identification and Control of Dynamical systems using NN”, IEEE Transactions on NN, v 1, n 1, pp 4-27, 1990.

10. Hornik, K., and Stinchcombe, "Multilayer feed forward networks are universal approximators, Neural Networks", n 2, pp 359-366, 1989.

11. James R. Muir, Clive Brereton, John R. Grace and C. Jim Lim, "Dynamic Modeling for Simulation and Control of a Circulating Fluidized Bed Combustor”, AIChE, v 43, n 5, pp 1141-1152.

12. Bhat, N., Minderman, P., and Jr. Thomas McAvoy, "Modeling chemical process systems via neural computation", IEEE Control Systems Magazine, v 10, pp 24-25, 1990.

13. Hunt, K.J., and Sbarbaro, D., "Neural networks for non-linear internal model control", IEEE Proceedings-D, v 138, n 5, pp 431-438, 1991.

14. Rao Vemuri, Artificial neural networks : concepts and control applications, Los Alamitos, Calif. : IEEE Computer Society Press, 1992

15. Handbook of intelligent control : neural, fuzzy, and adaptive approaches edited by David A. White, Donald A. Sofge, New York : Van Nostrand Reinhold, c1992.

16. Neuro-control systems : theory and applications, edited by Madan M. Gupta, Dandina H. Rao, New York : IEEE Press, c1994

17. Thomas E. Quantrille and Y.A. Liu., Artificial intelligence in chemical engineering, San Diego : Academic Press, c1991